

Elicitation for Preferences Single Peaked on Trees

Palash Dey

Indian Institute of Science, Bangalore
palash@csa.iisc.ernet.in

Neeldhara Misra

Indian Institute of Technology, Gandhinagar
mail@neeldhara.com

April 18, 2016

Abstract

In multiagent systems, we often have a set of agents each of which have a preference ordering over a set of items and one would like to know these preference orderings for various tasks, for example, data analysis, preference aggregation, voting etc. However, we often have a large number of items which makes it impractical to ask the agents for their complete preference ordering. In such scenarios, we usually elicit these agents' preferences by asking (a hopefully small number of) comparison queries — asking an agent to compare two items. Prior works on preference elicitation focus on unrestricted domain and the domain of single peaked preferences and show that the preferences in single peaked domain can be elicited by much less number of queries compared to unrestricted domain. We extend this line of research and study preference elicitation for single peaked preferences on trees which is a strict superset of the domain of single peaked preferences. We show that the query complexity crucially depends on the number of leaves, the path cover number, and the distance from path of the underlying single peaked tree, whereas the other natural parameters like maximum degree, diameter, pathwidth do not play any direct role in determining query complexity. We then investigate the query complexity for finding a weak Condorcet winner for preferences single peaked on a tree and show that this task has much less query complexity than preference elicitation. Here again we observe that the number of leaves in the underlying single peaked tree and the path cover number of the tree influence the query complexity of the problem.

1 Introduction

In multiagent systems, we often have scenarios where agents have to arrive at a consensus when choosing between multiple options. Typically, the agents have preferences over a set of items, and the problem of aggregating these preferences in a suitable manner is one of the most well-studied problems in social choice theory [BCE⁺15]. There are many ways of expressing preferences over a set of alternatives. One of the most comprehensive ways is to specify a complete ranking over the set of alternatives. However, one of the downsides of this model is the fact that it can be expensive to solicit a preference when there are a large number of alternatives, and many agents are involved.

Since asking agents to provide their complete rankings is impractical, a popular notion is one of *elicitation*, where we ask agents simple *comparison queries*, such as if they prefer alternative X over Y . This naturally gives rise to the problem of *preference elicitation*, where we hope to recover the complete ranking (or possibly the most relevant part of the ranking) based on a small number of queries.

The paradigm of *voting* is a popular general methodology for aggregating preferences, where one devises “voting rules” for mapping a collection of preferences (which we refer to as votes) to either a *winning alternative* or a *consensus ranking*. Keeping in line with the terminology used in voting, we will refer to alternatives as candidates, and a collection of votes will be termed a preference profile. In the context of a fixed voting rule, we may also want to query the voters up to the point of determining the winner (or the aggregate ranking, as the case may be). Yet another refinement in this setting is when we have prior information about how agents are likely to vote, and we may want to determine which voters to query first, to be able to quickly rule out a large number of alternatives, as explored by [CS02].

When our goal is to elicit preferences that have no prior structure, one can demonstrate scenarios where it is imperative to ask each agent (almost) as many queries as would be required to determine an arbitrary ranking. However, in recent times, there has been considerable interest in voting profiles that are endowed with additional structure. The motivation for this is two-fold. The first is that in several application scenarios commonly considered, it is rare that votes are ad-hoc, demonstrating no patterns whatsoever. For example, the notion of *single-peaked preferences*, which we will soon discuss at length, forms the basis of several studies in the analytical political sciences [HM97]. In his work on eliciting preferences that demonstrate the “single-peaked” structure, [Con09] argues that the notion of single-peakedness is also a reasonable restriction in applications outside of the domain of political elections.

The second motivation for studying restricted preferences is somewhat more technical, but is just as compelling. To understand why structured preferences have received considerable attention from social choice theorists, we must first take a brief detour into some of the foundational ideas that have shaped the landscape of voting rules as we understand them today. As it turns out, the axiomatic approach of social choice involves defining certain “properties” that formally capture the quality of a voting rule. For example, we would not want a voting rule to be, informally speaking, a *dictatorship*, which would essentially mean that it discards all but one voter’s input. Unfortunately, a series of cornerstone results establish that it is impossible to devise voting rules which respect some of the simplest desirable properties. Indeed, the classic works of Arrow [Arr50] and Gibbard-Satterthwaite [Gib73, Sat75] show that there is no straight forward way to simultaneously deal with properties like voting paradoxes, strategy-proofness, nondictatorship, unanimity etc. We refer to [Mou91] for a more elaborate discussion. Making the matter worse, many classical voting rules turn out to be computationally intractable.

This brings us to the second reason for why structured preferences are an important consideration. The notion of single-peakedness that we mentioned earlier is an excellent illustration (we refer the reader to Section 2 for the formal definition). Introduced by [Bla48], it not only captures the essence of structure in political elections, but also turns out to be extremely conducive to many natural theoretical considerations. To begin with, one can devise voting rules that are “nice” with respect to several properties, when preferences are single-peaked. Further, they are structurally elegant from the view of winner determination, since they always admit a *weak Condorcet winner* — a candidate which is not defeated by any other candidate in pairwise election — thus working around the Condorcet paradox which is otherwise a prominent concern in the general scenario. In a landmark contribution, [BBHH15] show that several computational problems that are intractable in the general setting become polynomially solvable when we consider single-peaked preferences.

A natural question at this point is if the problem of elicitation becomes any easier — that is, if we can get away with fewer queries — by taking advantage of the structure provided by single-

peakedness. It turns out that the answer to this is in the affirmative, as shown in a detailed study by [Con09]. The definition of single-peakedness involves an ordering over the candidates (called the *harmonious ordering* by some authors). The work of [Con09] shows that $\mathcal{O}(mn)$ queries suffice, assuming either that the harmonious ordering is given, or one of the votes is known. Dey and Misra [DM16] show a query complexity bound of $\mathcal{O}(mn)$ for the domain of single crossing profiles and a large number of voters.

We now return to the theme of structural restrictions on preferences. As it turns out, the single peaked preference domain has subsequently been generalized to single peakedness on trees (roughly speaking, these are profiles that are single peaked on every path) [Dem82, Tri89]. This is a class that continues to exhibit many desirable properties of single peaked domains. For example, there always exists a weak Condorcet winner and further, many voting rules that are intractable in unrestricted domain are polynomial time computable if the underlying single peaked tree is “nice” [YCE13, PE16]. We note the class of profiles that are single peaked on trees are substantially more general than the class of single peaked preferences. Note that the latter is a special case since a path is, in particular, a tree. Our work here addresses the issue of elicitation on profiles that are single-peaked on trees, and can be seen as a significant generalization of the results in [Con09]. We now detail the specifics of our contributions.

Parameter	Upper Bound	Lower Bound
Path width (w)	$\mathcal{O}(mn \log m)$ Observation 1	$\Omega(mn \log m)$ even for $w = 1, \log m$ Corollary 3
Maximum degree (Δ)	$\mathcal{O}(mn \log m)$ Observation 1	$\Omega(mn \log m)$ even for $\Delta = 3, m - 1$ Corollary 4
Path cover number (k)	$\mathcal{O}(mn \log k)$ Theorem 2	$\Omega(mn \log k)$ Corollary 2
Number of leaves (ℓ)	$\mathcal{O}(mn \log \ell)$ Corollary 1	$\Omega(mn \log \ell)$ Theorem 4
Distance from path (d)	$\mathcal{O}(mn + nd \log d)$ Theorem 3	$\Omega(mn + nd \log d)$ Theorem 6
Diameter (ω)	$\mathcal{O}(mn \log m)$ Observation 1	$\Omega(mn \log m)$ even for $\omega = 2, m/2$ Corollary 5

Table 1: Summary of query complexity bounds for PREFERENCE ELICITATION.

1.1 Motivation

1.2 Our Contributions

We study the query complexity for preference elicitation when the preference profile is single peaked on a tree. We provide *tight* connections between various parameters of the underlying tree and the query complexity for preference elicitation. Our broad goal is to provide a suitable generalization of preference elicitation for single peaked profiles to profiles that are single peaked on trees. Therefore, we consider various ways of quantifying the “closeness” of a tree to a path, and reflect on how these measures might factor into the query complexity of an algorithm that is actively exploiting the underlying tree structure.

We summarize our results for preference elicitation in Table 1, where the readers will note that most of the parameters (except diameter) chosen are small constants (typically zero, one or two) when the tree under consideration is a path. Observe that in some cases — such as the number of leaves, or the path cover number — the dependence on the parameter is transparent (and we

recover the results of [Con09] as a special case), while in other cases, it is clear that the perspective provides no additional mileage (the non-trivial results here are the matching lower bounds).

In terms of technique, our strategy is to “scoop out the paths from the tree” and use known algorithms to efficiently elicit the preference on the parts of the trees that are paths. We then efficiently merge this information across the board, and that aspect of the algorithm varies depending on the parameter considered. The lower bounds typically come from trees that provide large “degrees of freedom” in reordering candidates, typically these are trees that don’t have too many long paths (such as stars). The arguments are often subtle but intuitive.

We then study the query complexity for finding a weak Condorcet winner of a preference profile which is single peaked on a tree. Here, we are able to show that a weak Condorcet winner can be found with far fewer queries than the corresponding elicitation problem. In particular, we establish that a weak Condorcet winner can be found using $\mathcal{O}(mn)$ many queries for profiles that are single peaked on trees [Theorem 7], and we also show that this bound is the best that we can hope for [Theorem 10]. We also consider the problem for the special case of single peaked profiles. While [Con09] showed that $\Omega(mn)$ queries are necessary to determine the *aggregate ranking*, we show that only $\mathcal{O}(n \log m)$ queries suffice if we are just interested in (one of the) weak Condorcet winners. Moreover, we show this bound is tight under the condition that the algorithm does not interleave queries to different voters [Theorem 11] (our algorithm indeed satisfies this condition).

Finally, expressing the query complexity for determining a weak Condorcet winner in terms of a measure of closeness to a path, we show an algorithm with query complexity $\mathcal{O}(nk \log m)$ where k is the path cover number of \mathcal{T} [Theorem 12] or the number of leaves in \mathcal{T} [Corollary 6]. We now elaborate further on our specific contributions for preference elicitation.

- We design novel algorithms for preference elicitation for profiles which are single peaked on a tree with ℓ leaves with query complexity $\mathcal{O}(mn \log \ell)$ [Corollary 1]. Moreover, we prove that there exists a tree \mathcal{T} with ℓ leaves such that any preference elicitation algorithm for profiles which are single peaked on tree \mathcal{T} has query complexity $\Omega(mn \log \ell)$ [Theorem 4]. We show similar results for the parameter path cover number of the underlying tree [Theorem 2 and Corollary 2]. We provide a preference elicitation algorithm with query complexity $\mathcal{O}(mn + nd \log d)$ for single peaked profiles on trees which can be made into a path by deleting at most d nodes [Theorem 3]. We show that our query complexity upper bound is tight up to constant factors [Theorem 6]. These results show that the query complexity tightly depends on the number of leaves, the path cover number, and the distance from path of the underlying tree.
- We then show that there exists a tree \mathcal{T} with pathwidth one or $\log m$ [Corollary 3] or maximum degree is 3 or $m - 1$ [Corollary 4] or diameter is 2 or $m/2$ [Corollary 5] such that any preference elicitation algorithm for single peaked profiles on the tree \mathcal{T} has query complexity $\Omega(mn \log m)$. These results show that the query complexity of preference elicitation does not directly depend on the parameters above.

We next study query complexity for finding a weak Condorcet winner for profiles which are single peaked on trees and we have the following results.

- We show that a weak Condorcet winner can be found using $\mathcal{O}(mn)$ many queries for profiles that are single peaked on trees [Theorem 7] which is better than the query complexity for preference elicitation. Moreover, we prove that this bound is tight in the sense that any

algorithm for finding a weak Condorcet winner for profiles that are single peaked on stars has query complexity $\Omega(mn)$ [Theorem 10].

- On the other hand, we can find a weak Condorcet winner using only $\mathcal{O}(n \log m)$ many queries for single peaked profiles [Theorem 8]. Moreover, we show this bound is tight under the condition that the algorithm does not interleave queries to different voters [Theorem 11] (our algorithm indeed satisfies this condition). For any arbitrary underlying single peaked tree \mathcal{T} , we provide an algorithm for finding a weak Condorcet winner with query complexity $\mathcal{O}(nk \log m)$ where k is the path cover number of \mathcal{T} [Theorem 12] or the number of leaves in \mathcal{T} [Corollary 6].

To summarize, we remark that our results non-trivially generalize earlier works on query complexity for preference elicitation in [Con09]. We believe revisiting the preference elicitation problem in the context of profiles that are single peaked on trees is timely, and that this work also provides fresh algorithmic and structural insights on the domain of preferences that are single peaked on trees.

1.3 Related Work

We have already mentioned the work of [Con09] addressing the question of eliciting preferences in single-peaked profiles, which is the closest predecessor to our work. Before this, Conitzer and Sandholm addressed the computational hardness for querying minimally for winner determination [CS02]. They also prove that one would need to make $\Omega(mn \log m)$ queries even to decide the winner for many commonly used voting rules [CS05] which matches with the trivial $\mathcal{O}(mn \log m)$ upper bound for preference elicitation in unrestricted domain based on sorting. Ding and Lin study preference elicitation under partial information setting and show interesting properties of what they call a deciding set of queries [DL13]. Lu and Boutilier provide empirical study of preference elicitation under probabilistic preference model [LB11b] and devise several novel heuristics which often work well in practice [LB11a].

The rest of the paper is organized as follows: We introduce the basic terminologies and formally define the problems in Section 2, present results for eliciting preference profile and finding a weak Condorcet winner in Section 3 and Section 4 respectively, and conclude with future works in Section 5.

2 Preliminaries

For a positive integer n , we denote the set $\{1, \dots, n\}$ by $[n]$. Suppose we have a set $\mathcal{V} = \{v_i : i \in [n]\}$ of n voters each of which has a preference $>_i, i \in [n]$, alternatively called *vote*, which is a complete order over a set $\mathcal{C} = \{c_j : j \in [m]\}$ of m candidates. We denote the set of preferences over \mathcal{C} by $\mathcal{L}(\mathcal{C})$. The tuple $(>_i)_{i \in [n]}$ of the preferences of all the voters is called a *profile*. If not mentioned otherwise, we use m and n to denote the number of candidates and the number of voters respectively. Let $> \in \mathcal{L}(\mathcal{C})$ be any complete order over \mathcal{C} . For a subset $\mathcal{X} \subseteq \mathcal{C}$ of candidates, we denote the restriction of an order $>$ to the subset of candidates \mathcal{X} by $>(\mathcal{X})$. We denote the restriction of a profile $\mathcal{P} = (>_i)_{i \in [n]}$ to \mathcal{X} by $\mathcal{P}(\mathcal{X}) = (>_i(\mathcal{X}))_{i \in [n]}$. We say that a candidate $x \in \mathcal{C}$ is placed at the i^{th} position of a preference $> \in \mathcal{L}(\mathcal{C})$ if x is preferred over all but exactly $(i - 1)$ candidates in $>$. Given an n voters profile $\mathcal{P} = (>_i)_{i \in [n]}$, a candidate x is called a *Condorcet winner* of \mathcal{P} if for every other

candidate y , a *strict* majority of the voters prefer x over y ; that is $|\{i \in [n] : x \succ_i y\}| > n/2$ for every $y \in \mathcal{C} \setminus \{x\}$. A candidate x is called a *weak Condorcet winner* of an n voters profile $\mathcal{P} = (\succ_i)_{i \in [n]}$ if there does not exist any other candidate y whom a strict majority of the voters prefer over x ; that is $|\{i \in [n] : y \succ_i x\}| \leq n/2$ for every $y \in \mathcal{C} \setminus \{x\}$. For an order $\succ \in \mathcal{L}(\mathcal{C})$, we denote the “reverse order” of \succ by \succleftarrow ; that is $\succleftarrow = \{x \succleftarrow y : x, y \in \mathcal{C} \text{ and } y \succ x\}$. A preference $\succ \in \mathcal{L}(\mathcal{C})$ over a set of candidates \mathcal{C} is called *single peaked* with respect to an order $\succ' \in \mathcal{L}(\mathcal{C})$ if, for every candidates $x, y \in \mathcal{C}$, we have $x \succ y$ whenever we have either $c \succ' x \succ' y$ or $y \succ' x \succ' c$, where $c \in \mathcal{C}$ is the candidate at the first position of \succ . A profile $\mathcal{P} = (\succ_i)_{i \in [n]}$ is called single peaked with respect to an order $\succ' \in \mathcal{L}(\mathcal{C})$ if \succ_i is single peaked with respect to \succ' for every $i \in [n]$. Notice that if a profile \mathcal{P} is single peaked with respect to an order $\succ' \in \mathcal{L}(\mathcal{C})$, then \mathcal{P} is also single peaked with respect to the order \succleftarrow' . Given a path $\mathcal{Q} = (x_1, x_2, \dots, x_\ell)$ from a vertex x_1 to another vertex x_ℓ in a tree \mathcal{T} , we define the order induced by the path \mathcal{Q} to be $x_1 \succ x_2 \succ \dots \succ x_\ell$. Given a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ with the set of nodes as the set of candidates \mathcal{C} , a profile \mathcal{P} is called single peaked on the tree \mathcal{T} if \mathcal{P} is single peaked on the order induced by every path of the tree \mathcal{T} ; that is for every two candidates $x, y \in \mathcal{C}$, the profile $\mathcal{P}(\mathcal{X})$ is single peaked with respect to the order \succ of the candidates \mathcal{X} induced by the unique path from x to y in \mathcal{T} . We call the tree \mathcal{T} the *underlying single peaked tree*. It is known (c.f. [Dem82]) that there always exists a weakly Condorcet winner for a profile \mathcal{P} which is single peaked on a tree \mathcal{T} .

Trees. The following definitions pertaining to the structural aspects of trees will be useful.

- The *pathwidth* of \mathcal{T} is the minimum width of a *path decomposition* of \mathcal{T} [Hei93].
- A set of disjoint paths $\mathcal{Q} = \{Q_1 = (\mathcal{X}_1, \mathcal{E}_1), \dots, Q_k = (\mathcal{X}_k, \mathcal{E}_k)\}$ is said to cover a tree $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ if $\mathcal{X} = \cup_{i \in [k]} \mathcal{X}_i$, $\mathcal{E}_i \subseteq \mathcal{E}$, $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset$, $\mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ for every $i, j \in [k]$ with $i \neq j$. The *path cover number* of \mathcal{T} is the cardinality of the smallest set \mathcal{Q} of *disjoint* paths that cover \mathcal{T} .
- The *distance of a tree \mathcal{T} from a path* is the smallest number of nodes whose removal makes the tree a path.
- The *diameter* of a tree \mathcal{T} is the number of edges in the longest path in \mathcal{T} .

We also list some definitions of *subclasses* of trees (which are special types of trees, see also Figure 1).

- A tree is a *star* if there is a *center vertex* and every other vertex is a neighbor of this vertex.
- A tree is a *subdivision of a star* if it can be constructed by replacing each edge of a star by a path.
- A subdivision of a star is called *balanced* if there exists an integer ℓ such that the distance of every leaf node from the center is ℓ .
- A tree is a *caterpillar* if there is a *central path* and every other vertex is at a distance of one from it.
- A tree is a *complete binary tree* rooted at r if every nonleaf node has exactly two children and there exists an integer h , called the height of the tree, such that every leaf node is at a distance of either h or $h - 1$ from the root node r .

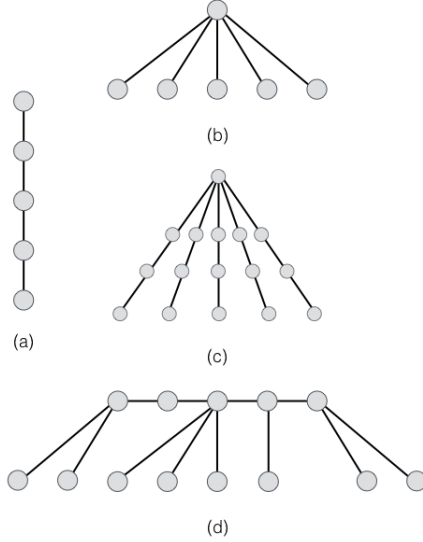


Figure 1: Depicting classes of trees: (a) a path, (b) a star, (c) a (balanced) subdivision of a star, (d) a caterpillar.

Problem Definitions and Known Results. Suppose we have a profile \mathcal{P} with n voters and m candidates. For any pair of distinct candidates x and y , and a voter $\ell \in [n]$, we introduce the boolean-valued function $\text{QUERY}(x >_{\ell} y)$. The output of this function is TRUE if the voter ℓ prefers the candidate x over the candidate y and FALSE otherwise. We now formally state the two problems that we consider in this paper.

Definition 1 PREFERENCE ELICITATION

Given a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ and an oracle access to the function $\text{QUERY}(\cdot)$ for a profile \mathcal{P} which is single peaked on \mathcal{T} , find \mathcal{P} .

Definition 2 WEAK CONDORCET WINNER

Given a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ and an oracle access to the function $\text{QUERY}(\cdot)$ for a profile \mathcal{P} which is single peaked on \mathcal{T} , find a weak Condorcet winner of \mathcal{P} .

Suppose we have a set of candidates $\mathcal{C} = \{c_1, \dots, c_m\}$. We say that an algorithm \mathcal{A} makes q queries if there are exactly q distinct tuples $(\ell, c_i, c_j) \in [n] \times \mathcal{C} \times \mathcal{C}$ with $i < j$ such that \mathcal{A} calls $\text{QUERY}(c_i >_{\ell} c_j)$ or $\text{QUERY}(c_j >_{\ell} c_i)$. We call the number of queries made by an algorithm \mathcal{A} its *query complexity*.

We state some known results that we will appeal to later. The first observation employs a sorting algorithm like merge sort to elicit every vote with $\mathcal{O}(m \log m)$ queries, while the second follows from the linear-time merge subroutine of merge sort ([Cor09]).

Observation 1 *There is a PREFERENCE ELICITATION algorithm with query complexity $\mathcal{O}(mn \log m)$.*

Observation 2 *Suppose $\mathcal{C}_1, \mathcal{C}_2 \subseteq \mathcal{C}$ form a partition of \mathcal{C} and $>$ is a ranking of the candidates in \mathcal{C} . Then there is a polynomial time algorithm that finds $>$ given $>(\mathcal{C}_1)$ and $>(\mathcal{C}_2)$ with query complexity $\mathcal{O}(|\mathcal{C}|)$.*

Theorem 1 [*Con09*] *There is a PREFERENCE ELICITATION algorithm with query complexity $\mathcal{O}(mn)$ for single peaked profiles.*

We now state the WEAK CONDORCET WINNER problem, which asks for eliciting only up to the point of determining a weak Condorcet winner (recall that at least one such winner is guaranteed to exist on profiles that are single-peaked on trees).

Definition 3 WEAK CONDORCET WINNER

Given a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ and an oracle access to the function $\text{QUERY}(\cdot)$ for a profile \mathcal{P} which is single peaked on \mathcal{T} , find a weak Condorcet winner of \mathcal{P} .

3 Results for Preference Elicitation

In this section, we present our results for PREFERENCE ELICITATION for profiles that are single peaked on trees. Recall that we would like to generalize Theorem 1 in a way to profiles that are single peaked on trees. Since the usual single peaked profiles can be viewed as profiles single peaked with respect to a path, we propose the following measures of how much a tree resembles a path.

- *Leaves.* Recall any tree has at least two leaves, and paths are the trees that have exactly two leaves. We consider the class of trees that have ℓ leaves, and show an algorithm with a query complexity of $\mathcal{O}(mn \log \ell)$.
- *Path Cover.* Consider the notion of a *path cover number* of a tree, which is the smallest number of disjoint paths that the tree can be partitioned into. Clearly, the path cover number of a path is one; and for trees that can be covered with k paths, we show an algorithm with query complexity $\mathcal{O}(mn \log k)$.
- *Distance from Paths.* Let d be the size of the smallest set of vertices whose removal makes the tree a path. Again, if the tree is a path, then the said set is simply the empty set. For trees that are at a distance d from being a path (in the sense of vertex deletion), we provide an algorithm with query complexity $\mathcal{O}(mn \log d)$.
- *Pathwidth and Maximum Degree.* Finally, we note that paths are also trees that have pathwidth one, and maximum degree two. These perspectives turn out to be less useful: in particular, there are trees where these parameters are constant, for which we show that elicitation is as hard as it would be on an arbitrary profile, and therefore the easy algorithm from Observation 1 is actually the best that we can hope for.

For the first three perspectives that we employ, that seemingly capture an appropriate aspect of paths and carry it forward to trees, the query complexities that we obtain are tight — we have matching lower bounds in all cases. Also, while considering structural parameters, it is natural to wonder if there is a class of trees that are incomparable with paths but effective for elicitation. Our attempt in this direction is to consider trees of bounded diameter. However, again, we find that this is not useful, as we have examples to show that there exist trees of diameter two that are as hard to elicit as general profiles.

We remark at this point that all these parameters are polynomially computable for trees, making the algorithmic results viable. Also, for the parameters of pathwidth, maximum degree and

diameter, we show lower bounds on trees where these parameters are large (such as trees with path-width $O(\log m)$, maximum degree $m - 1$, and diameter $m/2$), which — roughly speaking — also rules out the possibility of getting a good inverse dependence. As a concrete example, motivated by the $\mathcal{O}(mn)$ algorithm for paths, which have diameter m , one might wonder if there is an algorithm with query complexity $\mathcal{O}(\frac{mn \log m}{\log \omega})$. This possibility, in particular, is ruled out. We are now ready to discuss the results in Table 1.

We next show a structural result about trees namely any tree with ℓ leaves can be partitioned into ℓ paths. The idea is to fix some nonleaf node as root and iteratively find a path from low depth nodes (depth of a node is its distance from root) to some leaf node which is disjoint from all the paths chosen so far. We formalize this idea below.

Lemma 1 *Let $\mathcal{T} = (\mathcal{X}, \mathcal{E})$ be a tree with ℓ leaves. Then there is a polynomial time algorithm which partitions \mathcal{T} into ℓ disjoint paths $\mathcal{Q}_i = (\mathcal{X}_i, \mathcal{E}_i), i \in [\ell]$; that is we have $\mathcal{X}_i \cap \mathcal{X}_j = \emptyset, \mathcal{E}_i \cap \mathcal{E}_j = \emptyset$ for every $i, j \in [\ell]$ with $i \neq j$, $\mathcal{X} = \cup_{i \in [\ell]} \mathcal{X}_i$, $\mathcal{E} = \cup_{i \in [\ell]} \mathcal{E}_i$, and \mathcal{Q}_i is a path in \mathcal{T} .*

Proof: We first make the tree \mathcal{T} rooted at any arbitrary nonleaf node r . We now partition the tree \mathcal{T} into paths iteratively as follows. Initially every node of the tree \mathcal{T} is *unmarked* and the set of paths \mathcal{Q} we have is empty. Let $\mathcal{Q}_1 = (\mathcal{X}_1, \mathcal{E}_1)$ be the path in \mathcal{T} from the root node to any leaf node. We put \mathcal{Q}_1 into \mathcal{Q} and *mark* all the nodes in \mathcal{Q}_1 . More generally, in the i^{th} iteration we pick an unmarked node u that is closest to the root r , breaking ties arbitrarily, and add any path \mathcal{Q}_i in \mathcal{T} from u to any leaf node in the subtree \mathcal{T}_u rooted at u , and *mark* all the nodes in $\mathcal{Q}_i = (\mathcal{X}_i, \mathcal{E}_i)$. Since u is unmarked, we claim that every node in \mathcal{T}_u is unmarked. Indeed, otherwise suppose there is a node w in \mathcal{T}_u which is already marked. Then there exists two paths from r to w one including u and another avoiding u since u is currently unmarked and w is already marked. This contradicts the fact that \mathcal{T} is a tree. Hence every node in \mathcal{T}_u is unmarked. We continue until all the leaf nodes are marked and return \mathcal{Q} . Since \mathcal{T} has ℓ leaves and in every iteration at least one leaf node is marked (since every \mathcal{Q}_i contains a leaf node), the algorithm runs for at most ℓ iterations. Notice that, since the algorithm always picks a path consisting of unmarked vertices only, the set of paths in \mathcal{Q} are pairwise disjoint. We claim that \mathcal{Q} forms a partition of \mathcal{T} . Indeed, otherwise there must be a node x in \mathcal{T} that remains unmarked at the end. From the claim above, we have all the nodes in the subtree \mathcal{T}_x rooted at x unmarked which includes at least one leaf node of \mathcal{T} . This contradicts the fact that the algorithm terminates when all the leaf nodes are marked. \square

Using Lemma 1, and the fact that any path can account for at most two leaves, we have that the path cover number of a tree is same as the number of leaves up to a factor of two.

Lemma 2 *Suppose the path cover number of a tree \mathcal{T} with ℓ leaves is k . Then we have $\ell/2 \leq k \leq \ell$.*

Proof: The inequality $\ell/2 \leq k$ follows from the fact that any path in \mathcal{T} can involve at most two leaves in \mathcal{T} and there exists k paths covering all the leaf nodes. The inequality $k \leq \ell$ follows from the fact from Lemma 1 that any tree \mathcal{T} with ℓ leaves can be partitioned into ℓ paths. \square

3.1 Algorithmic Results

We now present our main algorithmic results. We begin with generalizing the result of Theorem 1 to any single peaked profiles on trees whose path cover number is at most k . The idea is to partition

the tree into k disjoint paths, use the algorithm from Theorem 1 on each paths to obtain an order of the candidates on each path of the partition, and finally merge these suborders intelligently. We now formalize this idea as follows.

Theorem 2 *There is a PREFERENCE ELICITATION algorithm with query complexity $\mathcal{O}(mn \log k)$ for profiles that are single peaked on trees with path cover number at most k .*

Proof: Since the path cover number is at most k , We can partition the tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ into t disjoint paths $\mathcal{P}_i = (\mathcal{C}_i, \mathcal{E}_i), i \in [t]$, where t is at most k . We now show that we can elicit any preference $>$ which is single peaked on the tree \mathcal{T} by making $\mathcal{O}(m \log t)$ queries which in turn proves the statement. We first find the preference ordering restricted to \mathcal{C}_i using Theorem 1 by making $\mathcal{O}(|\mathcal{C}_i|)$ queries for every $i \in [t]$. This step needs $\sum_{i \in [t]} \mathcal{O}(|\mathcal{C}_i|) = \mathcal{O}(m)$ queries since $\mathcal{C}_i, i \in [t]$ forms a *partition* of \mathcal{C} . We next merge the t orders $>(\mathcal{C}_i), i \in [t]$, to obtain the complete preference $>$ by using a standard divide and conquer approach for t -way merging which makes $\mathcal{O}(m \log t)$ queries [HUA83]. Thus the query complexity of the algorithms is $\mathcal{O}(m + m \log t) = \mathcal{O}(m \log k)$. \square

Towards our results on leaves, we show that the path cover of a tree with ℓ leaves is at least $\lfloor \ell/2 \rfloor$ and at most ℓ . The lower bound is easy since every path can account for at most two leaves. The upper bound comes from a careful partitioning of the nodes following maximal paths from leaves up to a root and using a careful marking scheme. We defer the details of this argument to a full version. Using the fact that a tree with ℓ leaves can be partitioned into at most ℓ paths, we can use the algorithm from Theorem 2 to obtain the following bound in terms of leaves.

Initially we have t orders to merge. We arbitrarily pair the t orders into $\lceil t/2 \rceil$ pairs with at most one of them being singleton (when t is odd). By renaming, suppose the pairings are as follows: $(>(\mathcal{C}_{2i-1}), >(\mathcal{C}_{2i})), i \in [\lceil t/2 \rceil]$. Let us define $\mathcal{C}'_i = \mathcal{C}_{2i-1} \cup \mathcal{C}_{2i}$ for every $i \in [\lceil t/2 \rceil]$ and $\mathcal{C}'_{\lceil t/2 \rceil} = \mathcal{C}_t$ if t is an odd integer. We merge $>(\mathcal{C}_{2i-1})$ and $>(\mathcal{C}_{2i})$ to get $>(\mathcal{C}'_i)$ for every $i \in [\lceil t/2 \rceil]$ using Observation 2. The number queries the algorithm makes in this iteration is $\sum_{i \in [\lceil t/2 \rceil]} \mathcal{O}(|\mathcal{C}_{2i-1}| + |\mathcal{C}_{2i}|) = \mathcal{O}(m)$ since $(\mathcal{C}_i)_{i \in [t]}$ forms a partition of \mathcal{C} . At the end of the first iteration, we have $\lceil t/2 \rceil$ orders $>(\mathcal{C}'_i), i \in [\lceil t/2 \rceil]$ to merge to get $>$. The algorithm repeats the step above $\mathcal{O}(\log t)$ times to obtain $>$ and the query complexity of each iteration is $\mathcal{O}(m)$.

Corollary 1 *There is a PREFERENCE ELICITATION algorithm with query complexity $\mathcal{O}(mn \log \ell)$ for profiles that are single peaked on trees with at most ℓ leaves.*

Finally, if we are given a subset of vertices whose removal makes the given tree a path, then we have an elicitation algorithm that makes $\mathcal{O}(mn + nd \log d)$. As before, we can determine the ordering among the candidates on the path with $\mathcal{O}(m - d)$ queries, and we determine the ordering among the rest in $\mathcal{O}(d \log d)$ queries using Observation 1, and finally merge using Observation 2. This leads us to the following.

Theorem 3 *There is a PREFERENCE ELICITATION algorithm with query complexity $\mathcal{O}(mn + nd \log d)$ for profiles that are single peaked on trees with distance d from path.*

Proof: Let \mathcal{X} be the smallest set of nodes of a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ such that $\mathcal{T} \setminus \mathcal{X}$, the subgraph of \mathcal{T} after removal of the nodes in \mathcal{X} , is a path. We have $|\mathcal{X}| \leq d$. For any preference $>$, we make $\mathcal{O}(d \log d)$ queries to find $>(\mathcal{X})$ using Observation 1, make $\mathcal{O}(|\mathcal{C} \setminus \mathcal{X}|) = \mathcal{O}(m - d)$ queries to find $>(\mathcal{C} \setminus \mathcal{X})$ using Theorem 1, and finally make $\mathcal{O}(m)$ queries to find $>$ by merging $>(\mathcal{X})$ and $>(\mathcal{C} \setminus \mathcal{X})$ using Observation 2. This gives an overall query complexity of $\mathcal{O}(mn + nd \log d)$. \square

3.2 Lower Bounds

We now turn to the lower bounds. Our first result is based on a counting argument, showing that the query complexity in terms of the number of leaves, given by Corollary 1, is tight up to constant factors. Indeed, let us consider a subdivision of a star with ℓ leaves and let t denote the distance from the center, so that we have a total of $t\ell + 1$ vertices. One can show that if the candidates are written out in level order, the candidates that are distance i from the star can be ordered arbitrarily within this ordering. This tells us that the number of possible preferences $>$ that are single peaked on the tree \mathcal{T} is at least $(\ell!)^t$. We obtain the lower bound by using a decision tree argument, wherein we are able to show that it is always possible for an oracle answering the comparison queries to answer in such a way that the total space of possibilities decreases by at most a factor of half. Since the tree must entertain at least $(\ell!)^t$ leaves to account for all possibilities, we obtain the claimed lower bound.

Theorem 4 *Let $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ be a balanced subdivision of a star with ℓ leaves. Then any PREFERENCE ELICITATION algorithm for single peaked profiles on \mathcal{T} has query complexity $\Omega(mn \log \ell)$.*

Proof: Suppose the number of candidates m be $(t\ell + 1)$ for some integer t . Let c be the center of \mathcal{T} . We denote the shortest path distance between any two nodes $x, y \in \mathcal{C}$ in \mathcal{T} by $d(x, y)$. We consider the partition $(\mathcal{C}_0, \dots, \mathcal{C}_t)$ of the set of candidates \mathcal{C} where $\mathcal{C}_i = \{x \in \mathcal{C} : d(x, c) = i\}$. We claim that the preference $> = \pi_0 > \pi_1 > \dots > \pi_t$ of the set of candidates \mathcal{C} is single peaked on the tree \mathcal{T} where π_i is any arbitrary order of the candidates in \mathcal{C}_i for every $0 \leq i \leq t$. Indeed; consider any path $\mathcal{Q} = (\mathcal{X}, \mathcal{E}')$ in the tree \mathcal{T} . Let y be the candidate closest to c among the candidates in \mathcal{X} ; that is $y = \operatorname{argmin}_{x \in \mathcal{X}} d(x, c)$. Then clearly $>(\mathcal{X})$ is single peaked with respect to the path \mathcal{Q} having peak at y . We have $|\mathcal{C}_i| = \ell$ for every $i \in [t]$ and thus the number of possible preferences $>$ that are single peaked on the tree \mathcal{T} is at least $(\ell!)^t$.

Let \mathcal{A} be any PREFERENCE ELICITATION algorithm for single peaked profiles on the tree \mathcal{T} . We now describe our oracle to answer the queries that the algorithm \mathcal{A} makes. For every voter v , the oracle maintains the set \mathcal{R}_v of possible preferences of the voter v which is consistent with the answers to all the queries that the algorithm \mathcal{A} have already made for the voter v . At the beginning, we have $|\mathcal{R}_v| \geq (\ell!)^t$ for every voter v as argued above. Whenever the oracle receives a query on v for any two candidates x and y , it computes the numbers n_1 and n_2 of orders in \mathcal{R}_v which prefers x over y and y over x respectively; the oracle can compute the integers n_1 and n_2 since the oracle has infinite computational power. The oracle answers that the voter v prefers the candidate x over y if and only if $n_1 \geq n_2$ and updates the set \mathcal{R}_v accordingly. Hence, whenever the oracle is queried for a voter v , the size of the set \mathcal{R}_v decreases by a factor of at most two. On the other hand, we must have, from the correctness of the algorithm, \mathcal{R}_v to be a singleton set when the algorithm terminates for every voter v – otherwise there exists a voter v (whose corresponding \mathcal{R}_v is not singleton) for which there exist two possible preferences which are single peaked on the tree \mathcal{T} and are consistent with all the answers the oracle has given and thus the algorithm \mathcal{A} fails to output the preference of the voter v correctly. Hence every voter must be queried at least $\Omega(\log((\ell!)^t)) = \Omega(t\ell \log \ell) = \Omega(m \log \ell)$ times. \square

Since the path cover number of a subdivided star on ℓ leaves is at least $\ell/2$, we also have the following.

Corollary 2 *There exists a tree \mathcal{T} with path cover number k such that any PREFERENCE ELICITATION algorithm for single peaked profiles on \mathcal{T} has query complexity $\Omega(mn \log k)$.*

Mimicking the level order argument above on a generic tree with ℓ leaves, and using the connection between path cover and leaves, we obtain lower bounds that are functions of (n, ℓ) and (n, k) , as given below. This will be useful for our subsequent results.

Theorem 5 *Let $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ be any arbitrary tree with ℓ leaves and path cover number k . Then any PREFERENCE ELICITATION algorithm for single peaked profiles on \mathcal{T} has query complexity $\Omega(n\ell \log \ell)$ and $\Omega(nk \log k)$.*

Proof: Let \mathcal{X} be the set of leaves in \mathcal{T} . We choose any arbitrary nonleaf node r as the root of \mathcal{T} . We denote the shortest path distance between two candidates $x, y \in \mathcal{C}$ in the tree \mathcal{T} by $d(x, y)$. Let t be the maximum distance of a node from r in \mathcal{T} ; that is $t = \max_{y \in \mathcal{C} \setminus \mathcal{X}} d(r, y)$. We partition the candidates in $\mathcal{C} \setminus \mathcal{X}$ as $(\mathcal{C}_0, \mathcal{C}_1, \dots, \mathcal{C}_t)$ where $\mathcal{C}_i = \{y \in \mathcal{C} \setminus \mathcal{X} : d(r, y) = i\}$ for $0 \leq i \leq t$. We claim that the preference $\succ = \pi_0 > \pi_1 > \dots > \pi_t > \pi$ of the set of candidates \mathcal{C} is single peaked on the tree \mathcal{T} where π_i is any arbitrary order of the candidates in \mathcal{C}_i for every $0 \leq i \leq t$ and π is an arbitrary order of the candidates in $\mathcal{C} \setminus \mathcal{X}$. Indeed, otherwise consider any path $\mathcal{Q} = (\mathcal{Y}, \mathcal{E}')$ in the tree \mathcal{T} . Let y be the candidate closest to r among the candidates in \mathcal{Y} ; that is $y = \operatorname{argmin}_{x \in \mathcal{Y}} d(x, r)$. Then clearly $\succ(\mathcal{Y})$ is single peaked with respect to the path \mathcal{Q} having peak at y . We have the number of possible preferences \succ that are single peaked on the tree \mathcal{T} is at least $|\mathcal{X}|! = \ell!$. Again using the oracle same as in the proof of Theorem 4, we deduce that any PREFERENCE ELICITATION algorithm \mathcal{A} for profiles that are single peaked on the tree \mathcal{T} needs to make $\Omega(n\ell \log \ell)$ queries. The bound with respect to the path cover number now follows from Lemma 2. \square

The following results can be obtained simply by applying Theorem 5 on particular graphs. For instance, we use the fact that stars have $(m - 1)$ leaves and have pathwidth one to obtain the first part of Corollary 3, while appealing to complete binary trees that have $O(m)$ leaves and pathwidth $O(\log m)$ for the second part. These examples also work in the context of maximum degree, while for diameter we use stars and caterpillars with a central path of length $m/2$.

We next consider the parameter pathwidth of the underlying single peaked tree. We immediately get the following result for PREFERENCE ELICITATION on trees with pathwidths one or $\log m$ from Theorem 5 and the fact that the pathwidths of a star and a complete binary tree are one and $\log m$ respectively.

Corollary 3 *There exist two trees \mathcal{T} and \mathcal{T}' with pathwidths one and $\log m$ respectively such that any PREFERENCE ELICITATION algorithm for single peaked profiles on \mathcal{T} and \mathcal{T}' respectively has query complexity $\Omega(mn \log m)$.*

Corollary 4 *There exist two trees \mathcal{T} and \mathcal{T}' with maximum degree $\Delta = 3$ and $m - 1$ respectively such that any PREFERENCE ELICITATION algorithm for single peaked profiles on \mathcal{T} and \mathcal{T}' respectively has query complexity $\Omega(mn \log m)$.*

Proof: Using Theorem 5, we know that any PREFERENCE ELICITATION algorithm for profiles which are single peaked on a complete binary tree has query complexity $\Omega(mn \log m)$ since a complete binary tree has $\Omega(m)$ leaves. The result now follows from the fact that the maximum degree Δ of a node is three for any binary tree. The case of $\Delta = m - 1$ follows immediately from

Theorem 5 applied on stars. □

Corollary 5 *There exists two trees \mathcal{T} and \mathcal{T}' with diameters $\omega = 2$ and $\omega = m/2$ respectively such that any PREFERENCE ELICITATION algorithm for profiles which are single peaked on \mathcal{T} and \mathcal{T}' respectively has query complexity $\Omega(mn \log m)$.*

Proof: The $\omega = 2$ and $\omega = m/2$ cases follow from Theorem 5 applied on star and caterpillar graphs with a central path of length $m/2$ respectively. □

Our final result, which again follows from Theorem 5 applied of caterpillar graphs with a central path of length $m - d$, shows that the bound in Theorem 3 is tight.

Theorem 6 *For any integers m and d with $1 \leq d \leq m/4$, there exists a tree \mathcal{T} with distance d from path such that any PREFERENCE ELICITATION algorithm for profiles which are single peaked on \mathcal{T} has query complexity $\Omega(mn + nd \log d)$.*

Proof: Consider the caterpillar graph where the length of the central path \mathcal{Q} is $m - d$; there exists such a caterpillar graph since $d \leq m/4$. Consider the order $\succ = \pi \succ \sigma$ of the set of candidates \mathcal{C} where π is an order of the candidates in \mathcal{Q} which is single peaked on \mathcal{Q} and σ is any order of the candidates in $\mathcal{C} \setminus \mathcal{Q}$. Clearly, \succ is single peaked on the tree \mathcal{T} . Any elicitation algorithm \mathcal{A} needs to make $\Omega(m - d)$ queries involving only the candidates in \mathcal{Q} to elicit π due to [Con09] and $\Omega(d \log d)$ queries to elicit σ due to sorting lower bound for every voter. This proves the statement. □

4 Weak Condorcet Winner

We now show that we can find a weak Condorcet winner of profiles that are single peaked on trees using fewer queries than the number of queries needed to find the profile itself. First, note that if a Condorcet winner is guaranteed to exist, then it can be found using $\mathcal{O}(mn)$ queries — we pit an arbitrary pair of candidates x, y and use $\mathcal{O}(n)$ queries to determine if x defeats y . We push the winning candidate forward and repeat the procedure, clearly requiring at most m rounds. Now, if a profile is single peaked with respect to a tree, and there are an odd number of voters, then we have a Condorcet winner and the procedure that we just described would work. Otherwise, we simply find a Condorcet winner among the first $(n - 1)$ voters. It can be shown that such a winner is one of the weak Condorcet winners for the overall profile, and we therefore have the following upper bound.

We begin with the following general observation.

Observation 3 *Let \mathcal{P} be a profile where a Condorcet winner is guaranteed to exist. Then we can find the Condorcet winner of \mathcal{P} by making $\mathcal{O}(mn)$ queries.*

Proof: For any two candidates $x, y \in \mathcal{C}$ we find whether x defeats y or not by simply asking all the voters to compare x and y ; this takes $\mathcal{O}(n)$ queries. The algorithm maintains a set \mathcal{S} of candidates which are *potential* Condorcet winners. We initialize \mathcal{S} to \mathcal{C} . In each iteration we pick any two candidates $x, y \in \mathcal{S}$ from \mathcal{S} , remove x from \mathcal{S} if x does not defeat y and vice versa using

$\mathcal{O}(n)$ query complexity until \mathcal{S} is singleton. After at most $m - 1$ iterations, the set \mathcal{S} will be singleton and contain only Condorcet winner since we find a candidate which is not a Condorcet winner in every iteration and thus the size of the set \mathcal{S} decreases by at least one in every iteration. This gives a query complexity bound of $\mathcal{O}(mn)$. \square

Using Observation 3 we now develop a WEAK CONDORCET WINNER algorithm with query complexity $\mathcal{O}(mn)$ for profiles that are single peaked on trees.

Theorem 7 *There is a WEAK CONDORCET WINNER algorithm with query complexity $\mathcal{O}(mn)$ for single peaked profiles on trees.*

For the special case of single peaked profiles, we can do even better. Here we take advantage of the fact that a “median candidate” [MCWG⁺95] is guaranteed to be a weak Condorcet winner. We make $\mathcal{O}(\log m)$ queries per vote to find the candidates placed at the first position of all the votes using the algorithm in [Con09] and find a median candidate to show the following.

if a profile is single peaked (on a path), then there is a WEAK CONDORCET WINNER algorithm with query complexity $\mathcal{O}(n \log m)$ as shown below. Let us define the frequency $f(x)$ of a candidate $x \in \mathcal{C}$ to be the number of votes where x is placed at the first position. Then we know that a median candidate according to the single peaked ordering of the candidates along with their frequencies as defined above is a weak Condorcet winner for single peaked profiles [MCWG⁺95].

Theorem 8 *There is a WEAK CONDORCET WINNER algorithm with query complexity $\mathcal{O}(n \log m)$ for single peaked profiles (on a path).*

Proof: Let \mathcal{P} be a profile that is single peaked with respect to an ordering $\succ \in \mathcal{L}(\mathcal{C})$ of candidates. Then we find, for every voter v , the candidate the voter v places at the first position using $\mathcal{O}(\log m)$ queries using the algorithm in [Con09] and return a median candidate. \square

Proof: Let $\mathcal{P} = (\succ_i)_{i \in [n]}$ be a profile which is single peaked on a tree \mathcal{T} . If n is an odd integer, then we know that there exists a Condorcet winner in \mathcal{P} since no two candidates can tie and there always exists at least one weak Condorcet winner in every single peaked profile on trees. Hence, if n is an odd integer, then we use Observation 3 to find a weak Condorcet winner which is the Condorcet winner too. Hence let us now assume that n is an even integer. Notice that $\mathcal{P}_{-1} = (\succ_2, \dots, \succ_n)$ is also single peaked on \mathcal{T} and has an odd number of voters and thus has a Condorcet winner. We use Observation 3 to find the Condorcet winner c of \mathcal{P}_{-1} and output c as a weak Condorcet winner of \mathcal{P} . We claim that c is a weak Condorcet winner of \mathcal{P} . Indeed otherwise there exists a candidate x other than c who defeats c in \mathcal{P} . Since n is an even integer, x must defeat c by a margin of at least two (since all pairwise margins are even integers) in \mathcal{P} . But then x also defeats c by a margin of at least one in \mathcal{P}_{-1} . This contradicts the fact that c is the Condorcet winner of \mathcal{P}_{-1} . \square

The next result uses Theorem 8 on paths in a path cover eliminating the case of even number of voters by the idea of setting aside one voter that was used in Theorem 7.

Theorem 9 *Let \mathcal{T} be a tree with path cover number at most k . Then there is an algorithm for WEAK CONDORCET WINNER for profiles which are single peaked on \mathcal{T} with query complexity $\mathcal{O}(nk \log m)$.*

Recalling that the number of leaves bounds the path cover number, we have the following consequence.

Corollary 6 *Let \mathcal{T} be a tree with ℓ leaves. Then there is an algorithm for WEAK CONDORCET WINNER for profiles which are single peaked on \mathcal{T} with query complexity $\mathcal{O}(n\ell \log m)$.*

We now state the lower bounds pertaining to WEAK CONDORCET WINNER. First, we show that any algorithm for single peaked profiles on stars has query complexity $\Omega(mn)$, showing that the bound of Theorem 7 is tight.

Theorem 10 *Any WEAK CONDORCET WINNER algorithm for single peaked profiles on stars must have query complexity $\Omega(mn)$.*

Proof: Let \mathcal{T} be a star with center vertex c . We now design an oracle that will “force” any WEAK CONDORCET WINNER algorithm \mathcal{A} for single peaked profiles on \mathcal{T} to make $\Omega(mn)$ queries. For every voter v , the oracle maintains a set of “marked” candidates which can not be placed at the first position of the preference of v . Suppose the oracle receives a query to compare two candidates x and y for a voter ℓ . If the order between x and y for the voter ℓ follows from the answers the oracle has already provided to all the queries for the voter ℓ , then the oracle answers accordingly. Otherwise it answers $x >_\ell y$ if y is unmarked and marks y ; otherwise the oracle answers $y >_\ell x$ and marks x . Notice that the oracle marks at most one unmarked candidate every time it is queried. We now claim that there must be at least $n/10$ votes which have been queried at least $m/4$ times. If not, then there exists $n - n/10 = 9n/10$ votes each of which has at least $m - m/4 = 3m/4$ candidates unmarked. In such a scenario, there exists a constant N_0 such that for every $m, n > N_0$, we have at least two candidates x and y who are unmarked in at least $(\lfloor n/2 \rfloor + 1)$ votes each. Now if the algorithm outputs x , then we put y at the first position in at least $(\lfloor n/2 \rfloor + 1)$ votes and at the second position in the rest of the votes and this makes y the (unique) Condorcet winner. If the algorithm does not output x , then we put x at the first position in at least $(\lfloor n/2 \rfloor + 1)$ votes and at the second position in the rest of the votes and this makes x the (unique) Condorcet winner. Hence the algorithm fails to output correctly in both the cases contradicting the correctness of the algorithm. Also the resulting profile is single peaked on \mathcal{T} with center at y in the first case and at x in the second case. Therefore the algorithm \mathcal{A} must have query complexity $\Omega(mn)$. \square

Our concluding result uses an intricate adversary argument, and shows that the query complexity for WEAK CONDORCET WINNER for single peaked profiles in Theorem 8 is essentially optimal, provided that the queries to different voters are not interleaved, as is the case with our algorithm.

Theorem 11 *Any WEAK CONDORCET WINNER algorithm for single peaked profiles which does not interleave the queries to different voters has query complexity $\Omega(n \log m)$.*

Proof: Let a profile \mathcal{P} be single peaked with respect to the ordering of the candidates $\succ = c_1 > c_2 > \dots > c_m$. The oracle maintains two indices ℓ and r for every voter such that any candidate from $\{c_\ell, c_{\ell+1}, \dots, c_r\}$ can be placed at the first position of the preference of the voter v and still be consistent with all the answers provided by the oracle for v till now and single peaked with respect to \succ . The algorithm initializes ℓ to one and r to m for every voter. The oracle answers any query in such a way that maximizes the new value of $r - \ell$. More specifically, suppose the oracle receives a query to compare candidates c_i and c_j with $i < j$ for a voter v . If the ordering between c_i and c_j follows, by applying transitivity, from the answers to the queries that have already been made so far for this voter, then the oracle answers accordingly.

Otherwise the oracle answers as follows. If $i < \ell$, then the oracle answers that c_j is preferred over c_i ; else if $j > r$, then the oracle answers that c_i is preferred over c_j . Otherwise (that is when $\ell \leq i < j \leq r$), if $j - \ell > r - i$, then the oracle answers that c_i is preferred over c_j and changes r to j ; if $j - \ell \leq r - i$, then the oracle answers that c_j is preferred over c_i and changes ℓ to i . Hence whenever the oracle answers a query for a voter v , the value of $r - \ell$ for that voter v decreases by a factor of at most two. Suppose the election instance has an odd number of voters. Let \mathcal{V} be the set of voters. Now we claim that the first $\lfloor n/5 \rfloor$ voters must be queried $(\log m - 1)$ times each. Suppose not, then consider the first voter v' that is queried less than $(\log m - 1)$ times. Then there exist at least two candidates c_t and c_{t+1} each of which can be placed at the first position of the vote v' . The oracle fixes the candidates at the first positions of all the votes that have not been queried till v' is queried (and there are at least $\lfloor 4n/5 \rfloor$ such votes) in such a way that $\lfloor n/2 \rfloor$ voters in $\mathcal{V} \setminus \{v'\}$ places some candidate in the left of c_t at the first positions and $\lfloor n/2 \rfloor$ voters in $\mathcal{V} \setminus \{v'\}$ places some candidate in the right of c_{t+1} . If the algorithm outputs c_t as the Condorcet winner, then the oracle makes c_{t+1} the (unique) Condorcet winner by placing c_{t+1} at the top position of v' , because c_{t+1} is the unique median in this case. If the algorithm does not output c_t as the Condorcet winner, then the oracle makes c_t the (unique) Condorcet winner by placing c_t at the top position of v' , because c_t is the unique median in this case. Hence the algorithm fails to output correctly in both the cases thereby contradicting the correctness of the algorithm. \square

From **Theorem 7 and 8** we have the following result for any arbitrary tree.

Theorem 12 *Let \mathcal{T} be a tree with path cover number at most k . Then there is an algorithm for WEAK CONDORCET WINNER for profiles which are single peaked on \mathcal{T} with query complexity $\mathcal{O}(nk \log m)$.*

Proof: Let \mathcal{P} be the input profile and $\mathcal{Q}_i = (\mathcal{X}_i, \mathcal{E}_i)$ $i \in [t]$ be $t(\leq k)$ disjoint paths that cover the tree \mathcal{T} . Here again, if the number of voters is even, then we remove any arbitrary voter and the algorithm outputs the Condorcet winner of the rest of the votes. The correctness of this step follows from the proof of Theorem 7. Hence we assume, without loss of generality, that we have an odd number of voters. The algorithm proceeds in two stages. In the first stage, we find the Condorcet winner w_i of the profile $\mathcal{P}(\mathcal{X}_i)$ for every $i \in [t]$ using Theorem 8. The query complexity of this stage is $\mathcal{O}(n \sum_{i \in [t]} \log |\mathcal{X}_i|) = \mathcal{O}(nt \log(m/t))$. In the second stage, we find the Condorcet winner w of the profile $\mathcal{P}(\{w_i : i \in [t]\})$ using Theorem 7 and output w . The query complexity of the second stage is $\mathcal{O}(nt \log t)$. Hence the overall query complexity of the algorithm is $\mathcal{O}(nt \log(m/t)) + \mathcal{O}(nt \log t) = \mathcal{O}(nk \log m)$. \square

5 Conclusions and Future Work

We show algorithms for preference elicitation for profiles which are single peaked on trees. Moreover, we prove that the query complexity of our algorithms are optimal up to constant factors. We also show that we need to query fewer number of times than preference elicitation if we only want to find any weak Condorcet winner.

In this work we do not assume any partial information about the preferences. However, in many scenarios, we may have some knowledge about the profile. An interesting future direction

of research is to study how having some partial information helps us reduce query complexity of preference elicitation.

Acknowledgement

Palash Dey wishes to gratefully acknowledge support from Google India for providing him with a special fellowship for carrying out his doctoral work.

References

- [Arr50] Kenneth J Arrow. A difficulty in the concept of social welfare. *The Journal of Political Economy*, pages 328–346, 1950.
- [BBHH15] Felix Brandt, Markus Brill, Edith Hemaspaandra, and Lane A Hemaspaandra. Bypassing combinatorial protections: Polynomial-time algorithms for single-peaked electorates. *Journal of Artificial Intelligence Research (JAIR)*, pages 439–496, 2015.
- [BCE⁺15] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel Procaccia. Handbook of computational social choice, 2015.
- [Bla48] Duncan Black. On the rationale of group decision-making. *The Journal of Political Economy*, pages 23–34, 1948.
- [Con09] Vincent Conitzer. Eliciting single-peaked preferences using comparison queries. *Journal of Artificial Intelligence Research (JAIR)*, 35:161–191, 2009.
- [Cor09] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.
- [CS02] Vincent Conitzer and Tuomas Sandholm. Vote elicitation: Complexity and strategy-proofness. In *Eighteenth National Conference on Artificial Intelligence (AAAI)*, pages 392–397, 2002.
- [CS05] Vincent Conitzer and Tuomas Sandholm. Communication complexity of common voting rules. In *Proceedings of the 6th ACM conference on Electronic Commerce (EC)*, pages 78–87. ACM, 2005.
- [Dem82] Gabrielle Demange. Single-peaked orders on a tree. *Mathematical Social Sciences*, 3(4):389–396, 1982.
- [DL13] Ning Ding and Fangzhen Lin. Voting with partial information: what questions to ask? In *Proceedings of the 12th International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, pages 1237–1238. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
- [DM16] Palash Dey and Neeldhara Misra. Preference elicitation for single crossing domain. In *IJCAI 2016, Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.
- [Gib73] Allan Gibbard. Manipulation of voting schemes: a general result. *Econometrica: Journal of the Econometric Society*, pages 587–601, 1973.

- [Hei93] Katherine Heinrich. Path decomposition. *Le matematiche*, 47(2):241–258, 1993.
- [HM97] Melvin J Hinich and Michael C Munger. *Analytical politics*. Cambridge University Press, 1997.
- [HUA83] John E Hopcroft, Jeffrey David Ullman, and Alfred Vaino Aho. *Data structures and algorithms*, volume 175. Addison-Wesley Boston, MA, USA:, 1983.
- [LB11a] Tyler Lu and Craig Boutilier. Robust approximation and incremental elicitation in voting protocols. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 287–293, 2011.
- [LB11b] Tyler Lu and Craig Boutilier. Vote elicitation with probabilistic preference models: Empirical estimation and cost tradeoffs. In *Algorithmic Decision Theory*, pages 135–149. Springer, 2011.
- [MCWG⁺95] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. *Microeconomic theory*, volume 1. Oxford university press New York, 1995.
- [Mou91] Hervi Moulin. *Axioms of cooperative decision making*. Number 15. Cambridge University Press, 1991.
- [PE16] Dominik Peters and Edith Elkind. Preferences single-peaked on nice trees. 2016.
- [Sat75] Mark Allen Satterthwaite. Strategy-proofness and arrow’s conditions: Existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10(2):187–217, 1975.
- [Tri89] Michael A Trick. Recognizing single-peaked preferences on a tree. *Mathematical Social Sciences*, 17(3):329–334, 1989.
- [YCE13] Lan Yu, Hau Chan, and Edith Elkind. Multiwinner elections under preferences that are single-peaked on a tree. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence (IJCAI)*, pages 425–431. AAAI Press, 2013.